Defects-in-Depth: Analyzing the Integration of Effective Defenses against One-Day Exploits in Android Kernels

Lukas Maar, *Graz University of Technology;* Florian Draschbacher, *Graz University of Technology and A-SIT Austria, Graz;* Lukas Lamster and Stefan Mangard, *Graz University of Technology*

https://www.usenix.org/conference/usenixsecurity24/presentation/maar-defects

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: Defects-in-Depth: Analyzing the Integration of Effective Defenses against One-Day Exploits in Android Kernels

Lukas Maar
Graz University of Technology

Florian Draschbacher
Graz University of Technology and A-SIT Austria

Lukas Lamster
Graz University of Technology

Stefan Mangard
Graz University of Technology

A Artifact Appendix

A.1 Abstract

We systematically analyze publicly available one-day exploits targeting the Android kernel over the past three years. We then demonstrate that integrating defense-in-depth mechanisms from the mainline Android kernel could mitigate 84.6 % of these exploits. This percentage serves as the ground truth for how secure mobile devices could be if their kernels were up to date with these defense mechanisms enabled. In a subsequent analysis of 994 devices, we reveal that the level of security that is actually achieved is severely lacking compared to the mainline Android kernel. This achieved security varies significantly depending on the vendor, ranging from mitigating 28.8 % to 54.6 % of exploits, indicating a 4.62 to 2.95[1] times worse scenario than the mainline kernel.

The artifacts include the dataset of 994 devices as well as the fully automated approach to determining the security achieved by vendor-specific kernels. For our dataset, we include the extracted kernel binaries with `kallsyms` of the top 7 vendors, i.e., Samsung, Xiaomi, Oppo, Vivo, Huawei, Realme and Motorola, along with Google, OnePlus and Fairphone, representing more than 84 % of the global market. For the automated approach, we provide various Python and shell scripts to reproduce the results of our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifacts do not perform any destructive steps. Crucially, while we provide the dataset of kernel binaries and `kallsyms` vulnerable to one-days for the artifact evaluation, we do not open source these as they could be used in a malicious intent. We explicitly mark the dataset as restricted view access only to the reviewers via Zenodo.

[1] Factors of $\frac{1-0.288}{1-0.846}$ and $\frac{1-0.546}{1-0.846}$, respectively.

A.2.2 How to access

We provide the source code (github) of the automated approach to determining the achieved security of vendor-specific kernels, and the compressed dataset (zenodo about 45 GB). Crucially, the uncompressed dataset is about 130 GB.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

A Linux system with `python3` and the following packages installed: `argparse`, `csv`, `json`, `os`, `subprocess`, `re`, `numpy`, and `matplotlib`.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

1. Install `python3` with the above requirements.
2. Clone our github repository (github) into the `/repo/path` directory.
3. Download the compressed kernel binaries with `kallsyms` files for our 994 evaluated devices (zenodo).
4. Decompress the kernel binaries and the `kallsyms` files to the appropriate vendor directory, e.g. `tar cfvz google.tar.gz /repo/path/google/firmwares`.

A.3.2 Basic Test

1. Execute `basic_test.py` within `/repo/path`.

2. This script will output `[+] Basic test success` if all software requirements are installed and all kernel binaries and `kallsyms` files are unpacked.

A.4 Evaluation workflow

A.4.1 Major Claims

We provide artifacts verifying the following claims:

(C1): We demonstrate that the 994 devices we analyzed can, on average, prevent 15.2 of the 26 one-day exploits (as shown in Figure 9 and described in Section 5.3.1 Susceptibility). This is proven by **(E2)**.

(C2): We demonstrate that protection against one-day exploits is highly vendor dependent (as shown in Figures 10c-l and described in Section 5.3.1 Susceptibility per Vendor). This is proven by **(E3)**.

(C3): We demonstrate that the Android kernel version used by vendor-provided kernel varies (as showsn in Figure 11 and described in Section 5.3.1 Susceptibility per Kernel Version). This is proven by **(E4)**.

(C4): We demonstrate that the susceptibility depends on the Android kernel version (as shown in Figure 12 and described in Section 5.3.1 Susceptibility per Kernel Version). This is proven by **(E5)**.

A.4.2 Experiments

Before performing experiments **(E2-5)**, please perform experiment **(E1)**, where all scripts are executed in `/repo/path`.

(E1): Create files with one day's susceptibility [30 human-seconds + 30 computer-minutes]:
How to: Execute `./do_all.sh`.
Execution: This script internally executes our evaluation script (i.e., `evaluate.py`) with all of our vendor-specific configuration files (located in `./config`) needed for the following experiments.
Results: It outputs all the required csv files containing the susceptibility per device, per vendor and per kernel version. These files are stored in the `output` directory.

(E2): **How to:** Execute `./gen_average.py`.
Results: It outputs the simplified version of Figure 9.

(E3): **How to:** Execute `./gen_plot.py </path/to/susceptibility/file>`.
Execution: The `</path/to/susceptibility/file>` should be a csv file created in **(E1)**, e.g., `./gen_plot.py output/google_one_day_analysis.csv` outputs Figure 10j.
Results: It outputs the Figures 10c-l.

(E4): **How to:** Execute `./gen_kernel_version_heatmap.py`.
Results: It outputs the Figure 11.

(E5): **How to:** Execute `./gen_one_day_heatmap.py`.
Results: It outputs the Figure 12.

A.5 Notes on Reusability

We provide the kernel binaries and `kallsyms` dataset for artifact evaluation instead of the firmware images, because this dataset has a size of about 45 GB compressed and 130 GB uncompressed. On the other hand, the entire set of firmware images is several terabytes in size, making it much more difficult to make this set available.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.